(12)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1990	memorandum

4. TITLE AND SUBTITLE
An Efficient Correspondence Based Algorithm for 2D and 3D Model Based Recognition

5. FUNDING NUMBERS
DACA76-85-C-0010
N00014-85-K-0124

AUTHOR(S)
Thomas M. Breuel

PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Artificial Intelligence Laboratory
545 Technology Square
Cambridge, Massachusetts 02139

8. PERFORMING ORGANIZATION REPORT NUMBER
AIM 1259

SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Information Systems
Arlington, Virginia 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
None

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution of this document is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This paper presents a polynomial time algorithm (pruned correspondence search, PCS) with good average case performance for solving a wide class of geometric maximal matching problems, including the problem of recognizing 3D objects from a single 2D image. Given two finite sets of geometric features with error bounds and a polynomial time algorithm that determines the feasibility of individual matchings, it finds a maximal matching. The algorithm is based on a pruned depth-first search for correspondences. Pruning is accomplished by representing regions of search space that have already been explored using an "adjoint list" of correspondences between image and model points.

The PCS algorithm is connected with the geometry of the underlying recognition problem only through calls to a verification algorithm. The analysis of the PCS algorithm demonstrates clearly the effects of the various combinatorial and geometric constraints on the complexity of the recognition problem.

(con't. on back)

14. SUBJECT TERMS (key words)		15. NUMBER OF PAGES
vision	model based recognition	23
object recognition	interpretation tree	16. PRICE CODE
linear programming	feature correspondences	

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

(block 13 con,t.)

Efficient verification algorithms, based on a linear representation of location constraints, are given for the case of affine transformations among vector spaces and for the case of rigid 2D and 3D transformations with scale.

Among the known algorithms that solve the bounded error recognition problem exactly and completely, the PCS algorithm currently has the lowest complexity. Some preliminary experiments suggest that PCS is a practical algorithm. Its similarity to existing correspondence based algorithms means that a number of existing techniques for speedup can be incorporated into PCS to improve its performance.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

C.B.I.P. Memo 59
A.I. Lab Memo 1259

October 1990

An Efficient Correspondence Based Algorithm for 2D and 3D Model Based Recognition

Thomas M. Breuel

Abstract

This paper presents a polynomial time algorithm (pruned correspondence search, PCS) with good average case performance for solving a wide class of geometric maximal matching problems, including the problem of recognizing 3D objects from a single 2D image. Given two finite sets of geometric features with error bounds and a polynomial time algorithm that determines the feasibility of individual matchings, it finds a maximal matching. The algorithm is based on a pruned depth-first search for correspondences. Pruning is accomplished by representing regions of search space that have already been explored using an "adjoint list" of correspondences between image and model points.

The PCS algorithm is connected with the geometry of the underlying recognition problem only through calls to a verification algorithm. The analysis of the PCS algorithm demonstrates clearly the effects of the various combinatorial and geometric constraints on the complexity of the recognition problem.

Efficient verification algorithms, based on a linear representation of location constraints, are given for the case of affine transformations among vector spaces and for the case of rigid 2D and 3D transformations with scale.

Among the known algorithms that solve the bounded error recognition problem exactly and completely, the PCS algorithm currently has the lowest complexity. Some preliminary experiments suggest that PCS is a practical algorithm. Its similarity to existing correspondence based algorithms means that a number of existing techniques for speedup can be incorporated into PCS to improve its performance.

This paper describes research done within the Center for Biological Information Processing, in the Department of Brain and Cognitive Sciences, and at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. This research is sponsored by a grant from the Office of Naval Research (ONR), Cognitive and Neural Sciences Division; by the Artificial Intelligence Center of Hughes Aircraft Corporation; by the Alfred P. Sloan Foundation; by the National Science Foundation; by the Artificial Intelligence Center of Hughes Aircraft Corporation (S1-801534-2); and by the NATO Scientific Affairs Division (0403/87). Support for the A. I. Laboratory's artificial intelligence research is provided by the Advanced Research Projects Agency of the Department of Defense under Army contract DACA76-85-C-0010, and in part by ONR contract N00014-85-K-0124.

1 Introduction

This paper is concerned with an efficient algorithm for geometric object recognition from visual data. We will study geometric recognition under under a bounded error model: assume we are given a set of image points $\mathcal{B}$ (say, in $\mathbb{R}^2$) and a set of model points $\mathcal{M}$ (say, in $\mathbb{R}^3$); determine whether there exists a viewing transformation T that will map each model point to within a distance ϵ of an image point. A more general version of this problem is to determine the size of the largest subsets of image and model points that can be brought into correspondence and the transformation that does this. This is illustrated in Figure 1.

There are several reasons why bounded error models are interesting formalizations of geometric recognition and estimation problems. Often, bounded error models answer statistical questions more directly than other statistical methods–in many aplications. we are only concerned with the question whether some particular value does not deviate from a true value by more than some given bound; the exact distribution of the error within those bounds is of no concern. Bounded error models also tend to be more robust and easier to apply than estimation techniques based on more specific models of error. Finally, bounded error models allow us to carry out a formal analysis of the computational complexity of recognition algorithms; such an analysis can give us important clues about the complexity and behavior of other recognition algorithms. such as those based on least square methods or alignment.

Bounded error models have received considerable attention in recent years. Their study in visual recognition (e.g., Grimson and Lozano-Perez, 1983, Baird, 1985, Breuel, 1989, Breuel, 1990, Cass, 1990) has been preceded by bounded error models in control (see Norton, 1986, for an introduction), which ultimately culminated in the development of a polynomial time algorithm for the linear programming problem Khachian, 1979 (see Papadimitriou and Steiglitz, 1982, for a simple introduction).

Baird, 1985, has analyzed the the problem of 2D visual object recognition under a bounded error model. He showed that determining a transformation given a 1-1 correspondence (matching) between a subset of image points and a subset of model points subject to convex polygonal error bounds can be solved efficiently by using a linear programming algorithm. He then used a simple depth-first search algorithm to search for the maximal 1-1 correspondence and gave an average case analysis of the complexity of the search for random point patterns in the absence of spurious data or occlusions and assuming that a match actually exists. His algorithm empirically performs well even if a few model features are missing from the image and a few spurious features have been added to the image, but he could not bound the worst case running time of the algorithm with a polynomial. The chief advantage of his search technique is, however, that it extends in a straightforward

manner to higher dimensions. The search technique proposed by Grimson and Lozano-Perez, 1983, is similar to Baird's method: they have proposed pruning heuristics that improve the performance of their search algorithm in practice. But such heuristics have not yielded a provably polynomial time recognition algorithm (Grimson, 1988, Grimson, 1989).

Cass, 1990, has used a different approach to the same problem. He observes that the space of transformations is partitioned into a polynomial number of cells by correspondences between model and image features. The search for maximal 1-1 correspondences can then be restricted to examining the boundaries of these cells of transformation space. By using some additional constraints, Cass reduces the problem to a 1D problem, which means that the boundaries separating the cells are a (provably small) set of discrete points. This "topological" method has the advantage that it is easily demonstrated to require only a polynomial number of arithmetic operations in the 2D case. Cass' method is essentially a sweep of transformation space[1] (e.g., Edelsbrunner, 1987).

A very important contribution of Baird, 1985, was the use of a representation of transformations that ensured that linear constraints on the location of image features gave rise to linear constraints on the set of feasible transformations. Baird's linear representation lets us apply linear programming methods to the verification problem.

Based on this correspondence between linear constraints on location and halfspaces (linear constraints) in transformation space, the existence of a polynomial time algorithm for a wide class of recognition problems is easy to see: the linear constraints on feature locations will give rise to a partition of transformation space into a polynomial number of cells. Standard algorithms from computational geometry can be used for enumerating these cells. Each individual cell will correspond to a number of geometrically equivalent matchings between image and model points that can be determined by explicitly applying one of the transformations in the cell to the model and comparing the result with the image.

The performance of transformation space sweep algorithms in the average case is, however, disappointing: many cells enumerated by the transformation space sweep correspond only to small matchings. In practice, correspondence search algorithms can perform significantly better, even though the worst case complexity of previous correspondence search algorithms is provably exponential. The empirical efficiency of correspondence search algorithms is a consequence of the fact that they can use geometric constraints to eliminate most of the smaller geometrically inconsistent matchings.

This paper first describes the pruned correspondence search (PCS) algorithm, an algo-

[1] Somewhat similar in spirit to Cass, 1990, but more restricted in scope, is the result of Alt *et al.*, 1988, which applies only to the problem of proving the existence of a bounded matching between two sets of points under uniform, circular error bounds

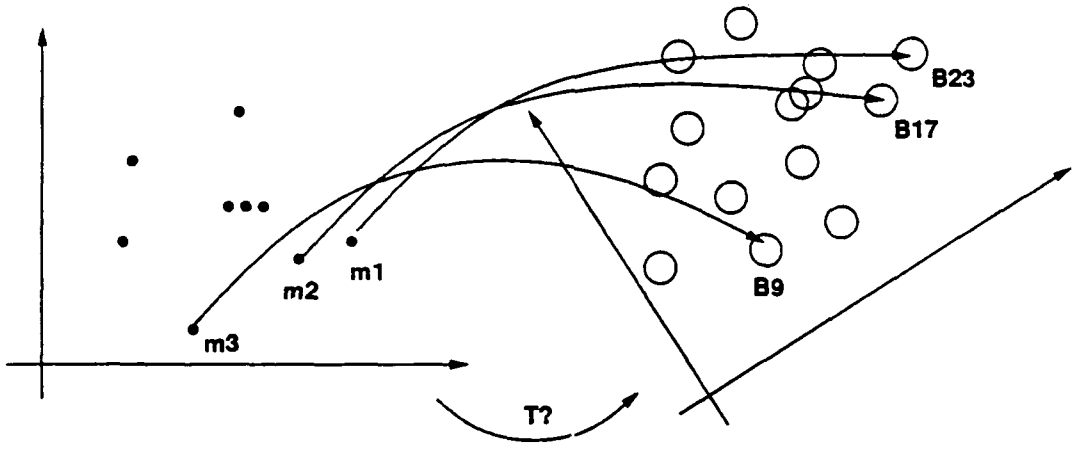

Figure 1: A formalization of the recognition problem with bounded error: Find the largest subset of points m_i on the left such that there exists a transformation T (translation, rotation, scale) of the plane that maps the points into the error bounds $B_j = b_j + E_j$ given by image points b_j together with error bounds given as sets E_j on the right.

rithm that combines the advantages of worst case polynomial time complexity with the good average case behavior of previous correspondence algorithms. The PCS algorithm consists of a polynomial time transformation of a verification algorithm into a recognition algorithm. The central idea underlying the PCS algorithm is that regions of transformation space that have already been explored can be represented efficiently and concisely using a list of pairings between image and model points.

In the second part, the paper describes how to apply the PCS algorithm to the problem of 2D and 3D model based recognition. In order to do so, we extend the linear formulation introduced by Baird to the non-linear problem of verifying 3D objects in 2D images under rigid transformations. Then, worst case and average case complexities for the algorithm applied to the problems of 2D and 3D recognition from 2D images are derived. It is also discussed how the PCS algorithm can be integrated into any existing correspondence search based algorithm with little additional overhead, taking advantage of existing heuristic methods while guaranteeing worst case polynomial time complexity even in the presence of large error bounds, clutter, and occlusions.

2 The Algorithm

Let us begin by stating the formal problem that the algorithm presented in this paper solves (following closely the definitions in Baird, 1985):

Definition 1 *Given a set of image points $\mathcal{B} = \{b_1, \ldots, b_k\}$, a set of model points $\mathcal{M} = \{m_1, \ldots, m_l\}$, a set of error bounds $\mathcal{E} = \{E_1, \ldots, E_k\}$ and a set of permissible transformations $\mathcal{T}$, a* MATCHING[2] *consists of two subsets $\mathcal{B}^\mu \subseteq \mathcal{B}$ and $\mathcal{M}^\mu \subseteq \mathcal{M}$, and a permutation μ. A matching is* FEASIBLE *if there exists some transformation $T \in \mathcal{T}$ such that*[3] *$T\, b_i \in m_{\mu(i)} + E_i$, for all $b_i \in \mathcal{B}^\mu$.*

The SIZE *of a matching μ (written as* size(μ)*) is the size of $\mathcal{B}^\mu$ (or, equivalently, $\mathcal{M}^\mu$).*

A matching can be viewed as a collection of PAIRINGS *or* CORRESPONDENCES*, where a pairing is a pair consisting of an image point and a set from the collection of model sets; in different words, a pairing is an element of $\mathcal{B} \times \mathcal{M}$.*

Sometimes we will call a pairing a CONSTRAINT *if we think of it as restricting the set of compatible transformations T.*

Definition 2 *The problem of verifying a bounded matching (VBM) is to determine whether, given a matching, there exists a transformation compatible with that matching.*

Definition 3 *The* MAXIMAL BOUNDED MATCHING *(MBM) problem is to find a feasible matching such that no other feasible matching is larger.*

In the definition of MBM, it is sufficient to exhibit any one of these maximal matchings, as long as no other matching is larger. More generally, we might wish the algorithm to return a concise description of *all* maximal matchings; the algorithm presented in this paper is capable of doing this. In fact, the algorithm will return a small set of representative maximal matchings, which could be used to reconstruct each possible maximal matchings efficiently.

An alternative definition places the error bounds on the model points. We will return briefly to this distinction later; for the time being it is irrelevant, since we only need to make the following assumptions about the bounded matching problem at hand.

[2]The definition of a "matching" follows Baird, 1985; it is not identical to the definition of a matching in graph theory. There are several terminologies in common use for talking about correspondence based recognition algorithms. In this paper, we will use the term "correspondence" when discussing classes of algorithms, and refer to individual correspondences as "pairings" and sets of correspondences as "matchings".

[3]The notation $v + E$, where v is a vector and E is a set, refers to the set $v + E = \{v + e : e \in E\}$.

We assume that there is a polynomial time algorithm, say, VBME (VERIFICATION of a BOUNDED MATCHING with EXCLUSION), that, given two sets of pairings $P = \{p_1, \ldots, p_n\}$ and $A = \{a_1, \ldots, a_n\}$ (as in Definition 1), can determine whether there exists a transformation T that is consistent with all the pairings in P and consistent with none of the pairings in A. For example, in the case of convex polygonal error bounds on 2D points, this can be done using linear programming, as we will see. Note that VBME may, in general, be a harder problem than VBM; for example, if the constraints imposed by a pairing on the set of feasible transformations are convex, VBM has to determine whether the intersection of a number of convex sets is empty, whereas VBME has to determine whether the intersection of a number of convex sets with one non-convex set is non-empty.

The combinatorial constraint that correspondence based algorithms rely on is the observation that if some matching P is infeasible (geometrically inconsistent), no matching $P' \supseteq P$ can be feasible. A natural approach to finding a maximal matching is therefore a depth first search: we start with an empty matching, and add pairings to it until the current matching becomes infeasible. We call a matching to which no other pairing can be added without making it infeasible a LEAF. The set of all leaves forms a set of candidates for maximal matchings.

Just terminating the search when a matching has become infeasible is, however, insufficient to guarantee polynomial time complexity of the search algorithm. The reason is that geometrically equivalent matchings may be re-explored by the search algorithm a large number of times. Consider the extreme example of matching n image points at the origin against n model points at the origin: in this case, there exist $n!$ different (as sets) maximal matchings, and all of these will be explored as leaves by a simple correspondence search algorithm.

However, knowing a single representative of this set of geometrically equivalent matchings is sufficient to recover all the other matchings by alignment, as follows. Given the representative matching μ, we compute a transformation T consistent with it. We can then use T to transform the model into the image and test whether any given other matching μ' is equivalent to μ directly.

It is therefore sufficient to return only a single representative for a set of geometrically equivalent matchings. From the above argument, we can also deduce what technique we can use in order to make sure that we only compute one of the possibly exponentially many equivalent matchings: all geometrically equivalent matchings correspond to the same set of transformations between image and model space. If we can keep track during the search of correspondence which regions of transformation space have already been explored, we can avoid the exponential behavior of the simple correspondence search algorithms.

The crucial idea presented in this paper is that the regions that have already been explored by the search algorithm can be represented concisely and efficiently as another set of

pairings, the ADJOINT PAIRINGS. *By putting a pairing of a given image point with a given model point on the adjoint list, we ensure that no transformations that map the model point to within the given error bounds of the image point will ever be reconsidered by the algorithm. This fact is the main motivation behind the PCS algorithm.*

An algorithm based on these ideas is given in the Scheme programming language in[4] Figure 3. The algorithm is implemented by the function `all-representatives`. This function generates a list of candidates which can then be tested for maximality later. The function `all-representatives` takes as arguments a function `vbme` implementing the verification with exclusion algorithm and a list of all possible pairings. The function `complement` complements a list of pairings with respect to the list of all pairings `pairings`. Other functions will be explained in the proof of the correctness and polynomial time complexity of the algorithm. The pruning method used by the algorithm is illustrated schematically in Figure 2.

The pruning step based on the adjoint list of pairings is relatively low cost, and it can only reduce the number of nodes expanded during the search. It turns out that under weak assumptions, this pruning step is actually sufficient to guarantee polynomial time behavior of a correspondence search algorithm. Below, we will present a proof of this fact based on the geometrical view of transformation space described in, for example, Grimson and Huttenlocher, 1989, and Cass, 1990, which makes it easy to relate the results about the complexity of the PCS algorithm to transformation space based methods[5] (sweeps and sampling methods).

Consider a pairing between a model point m_i, an image point b_j, and a set E_j specifying the error bounds. Corresponding to this choice is a set of transformations $\mathcal{T}_{ij} = \{T : T\, m_i \in b_j + E_j\}$, i.e., the set of transformations that will map the image point to within the error bound of a prototype model[6].

The set $\mathcal{L}$ of all intersections and unions of the $\mathcal{T}_{ij}$ forms a lattice under inclusion. The set of lower bounds of $\mathcal{L} - \{\emptyset\}$ form a PARTITION $\mathcal{P}$ of transformation space[7]. We will refer to the individual elements of $\mathcal{P}$ as CELLS. A REPRESENTATIVE for a cell is a set of

[4]See Rees and Clinger, 1986, for a description of the Scheme programming language. The functions **`remove-if-not`** and **`some`** are defined like their CommonLisp counterparts. The functions **`pairing-from`** and **`pairing-to`** return some unique label for the image or model point, respectively, that constitute the pairing.

[5]An alternative proof of the polynomial time complexity of the PCS algorithm based on image space is possible and yields sharper worst case complexity bounds for important special cases of recognition under the bounded error model.

[6]To avoid discussion of some unimportant special cases, we will assume that different pairings give rise to non-identical constraints in transformation space; i.e., $i \neq i' \vee j \neq j' \Rightarrow \mathcal{T}_{ij} \neq \mathcal{T}_{i'j'}$.

[7]Without loss of generality, we assume that $\mathcal{T} = \bigcup_{ij} \mathcal{T}_{ij}$. The elements of this partition are unions of the elements of the arrangement (see Edelsbrunner, 1987) generated by the hyperplanes containing the faces of the convex polyhedra in transformation space.

pairings P such that the set of transformations compatible with all the pairings in P is just the transformations in the cell.

For example, in the case studied by Baird, the individual constraints correspond to halfspaces in transformation space; each $\mathcal{T}_{ij}$ consists of the intersection of a number of such halfspaces (i.e., is itself a convex polyhedron in transformation space). In the case studied by Cass, each $\mathcal{T}_{ij}$ can be represented by a generalized cylindrical shape in[8] $\mathbb{R}^2 \times C^1$.

In general, the size of $\mathcal{P}$ (i.e., $^{\#}\mathcal{P}$) could be exponential in the number of image and model features. However, for many classes of transformations and image/model spaces, $^{\#}\mathcal{P}$ is only polynomial in size. We will examine conditions for this later; in the case of linear constraints (convex polygonal error bounds), $\mathcal{P}$ is composed of unions of elements of the arrangement of hyperplanes corresponding to the linear constraints on feature locations.

We need some additional notation for the proof. Imagine that there exists a function *consistent*(T, P) that returns **true** iff the transformation T is consistent with every pairing $p \in P$. P, Q, R will refer to sets of pairings (some matching), and A, B, C refer to sets of adjoint pairings, i.e., represent regions of transformation space excluded from further search. Assume that all possible pairings are numbered and list them as $\{q_1, \ldots, q_n\}$. Also, we introduce the following abbreviations for certain subsets of transformation space:

Definition 4

$$\mathcal{T}(P) = \{T : \forall p \in P : \mathit{consistent}(T, \{p\})\}$$

$$\mathcal{T}(P, A) = \{T : \forall p \in P : \mathit{consistent}(T, \{p\}), \forall a \in A : \neg \mathit{consistent}(T, \{a\})\}$$

$$\mathcal{D}_i^P = \mathcal{T}(\{p_i\}, \{p_1, \ldots, p_{i-1}\}) \qquad \text{where} \qquad P = p_1, \ldots, p_i$$

Now, let us examine the algorithm in more detail (refer to Figure 3). In the discussion of the algorithm, we will have to refer to functions in the Scheme code; **typewriter** font and standard mathematical notation will be used in such cases; e.g., the Scheme call (`leaf p`) will be written as `leaf`(P) in proofs.

Let us first consider some simple lemmas:

Lemma 1 *The sets* $\mathcal{T}(P, A)$ *and* $\mathcal{D}_i^P$, *where* $P = \{p_1, \ldots, p_n\}$ *are given by:*

$$\mathcal{T}(P, A) = \bigcap_{p_i \in P} \mathcal{T}_i - \bigcup_{q_i \in A} \mathcal{T}_i \qquad \text{and} \qquad \mathcal{D}_i^P = \mathcal{T}(p_i) - \bigcup_{j=1}^{i-1} \mathcal{T}(p_j)$$

Proof. Follows directly from the definition. □

[8] C^1 is the circle.

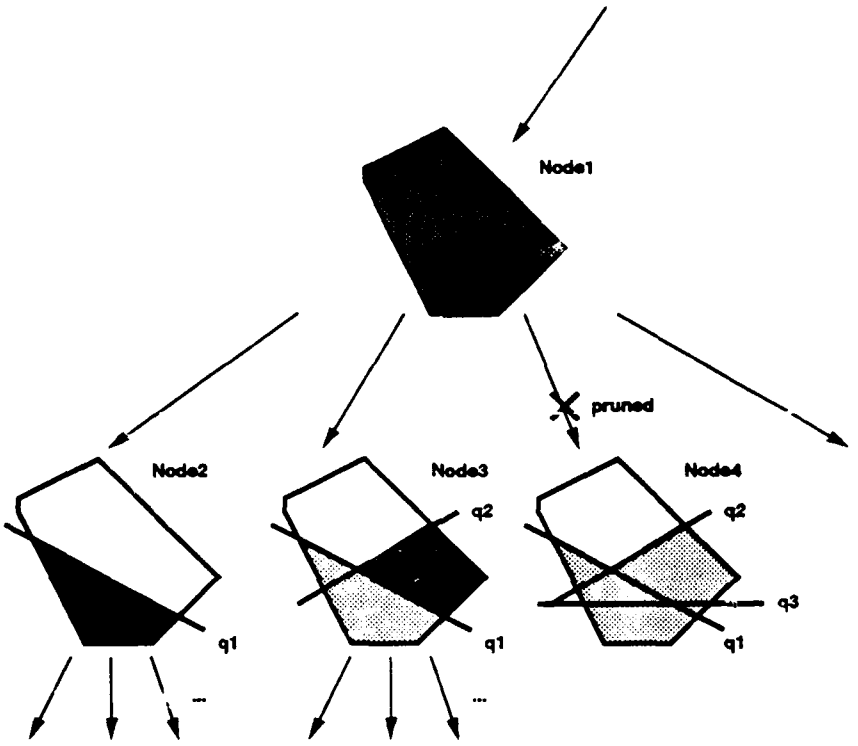

Figure 2: This figure illustrates schematically how the search tree for a maximal matching is pruned. The polygon indicates the region in transformation space that is compatible with the set of pairings P_1 at Node 1 and not excluded by the adjoint constraints A_1 at the same node.

The region of transformation space that a particular node is responsible for exploring is shown in dark grey. Node 2 is responsible for exploring the region where all constraints in $P_1 \cup \{q_1\}$ are satisfied. Node 3 does not need to explore this region anymore: therefore, q_1 has been added to the adjoint constraints when Node 3 is expanded.

The search tree starting at Node 4 can be pruned in this example, because all the matchings containing q_3 will already have been explored by nodes 2 and 3. Notice that this is a geometrical fact, not a combinatorial fact–if q_3 had been a little higher so that it included some of the unexplored–white–region, this branch could not have been pruned from the search tree.

```
(define (all-representatives vbme pairings)
  ;; all the elements in "pairings" that are not in "x"
  (define (complement x)
    (remove-if-not (lambda (y) (not (member y p))) pairings))

  ;; return a list of the form
  ;; (((q1.P) A) ((q2.P) (q1.A)) ((q3.P) (q1 q2.A)) ...)
  (define (successors p a)
    (define (successors1 q)
      (if (null q) '()
          (cons (list (cons (car q) p) (append (cdr q) a))
                (successors1 (cdr q)))))
    (remove-if-not (lambda (x) (apply vmbe x))
     (successors1 (complement x))))

  ;; determine whether "p" is a feasible matching and has no successors
  (define (leaf p)
    (and (vbme p '())
         (not (some (lambda (x) (vbme (cons x p) '())) pairings))))

  ;; recursively compute representatives for all leaves
  ;; that are candidates for being maximal matchings
  (define (representatives p a)
    (if (leaf p) (list p)
        (reduce append
          (map (lambda (x) (apply representatives1 x))
       (successors p a)))))

  (representatives '() '()))
```

Figure 3: The pruned search algorithm. See the text for an explanation.

The idea is that the $\mathcal{D}_i^P$ partition the part of transformation space that is of interest at any particular node during the search, and that these sets can be represented in the algorithm concisely and efficiently by a pairing p_i and a list of adjoint pairings $p_1, \ldots, p_{i-1}$ (in the algorithm itself, this is a pair consisting of two lists, $((p_i)\ (p_1\ \cdots\ p_{i-1}))$. We will therefore refer to calling **representatives**(P, A) as calling **representatives** "on the set" represented by the transformations compatible with the pairings P and adjoint pairings A.

Now, some more trivial lemmas:

Lemma 2 *The sets $\mathcal{D}_i^P$ partition the space of transformations accessible from the pairings in $P = \{p_1, \ldots, p_n\}$:*

$$\bigcup_i \mathcal{T}(p_i) = \bigcup_i \mathcal{D}_i^P \qquad i \neq j \Rightarrow \mathcal{D}_i^P \cap \mathcal{D}_j^P = \{\}$$

representatives(P, A) *returns only leaves:*

$$\forall Q \in \textbf{representatives}(P, A) : \textbf{leaf}(Q) = \textbf{true}$$

A leaf cannot be divided further by another pairing; a leaf is just a representation, in terms of a largest set of pairings, of a cell of the partition $\mathcal{P}$ of transformation space induced by the individual pairings:

$$(\textbf{leaf}(P) = \textbf{true}) \wedge (\mathcal{T}(P, \{\}) \cap \mathcal{T}_i \neq \emptyset) \Rightarrow \mathcal{T}(P, \{\}) \cap \mathcal{T}_i = \mathcal{T}(P, \emptyset)$$

Proof. Left to the reader. □

Lemma 3

$$\{\mathcal{T}(P) : P \in \textbf{representatives}(\{\}, \{\})\} = \{\mathcal{T}(Q) : \textbf{leaf}(Q)\}$$

Proof. What we are trying to show here is that **representatives** will return a representative for every leaf in transformation space[9].

Since **representatives** only returns leaves, "$\subseteq$" has been established.

To prove "$\supseteq$", we will demonstrate that if **leaf**(Q) is **true** and $\mathcal{T}(Q, A) \subseteq \mathcal{T}(P, A)$ then **representatives**(P, A) will return a list containing a representative of $\mathcal{T}(Q)$. This will prove, in particular, that **representatives**$(\{\}, \{\})$ must return all P, as required, since the empty set of pairings and the empty set of adjoint constraints correspond to all of transformation space.

If P is itself a leaf, then $\mathcal{T}(P) = \mathcal{T}(Q)$ and we are done.

[9]The statement of the lemma uses the cell $\mathcal{T}(P)$ in transformation space to emphasize that the representation of the matchings P and Q as sets or lists of pairings does not affect the result.

Otherwise, consider the set $R = \{r_1, \ldots, r_n\}$ of successor pairings at a search node. If we use no pruning, the search at this point would explore the intersection $\mathcal{T}'$ of the set of currently feasible transformations $\mathcal{T}(P, A)$ with the set of transformations that are accessible by the pairings in R. But the sets $\mathcal{D}'_k = \mathcal{T}' \cap \mathcal{D}_k^R$ form a partition of the set $\mathcal{T}'$, by Lemma 2. Thus, there must be a $\mathcal{D}'_k$ such that $\mathcal{T}(Q) \subseteq \mathcal{D}'_k$. But `representatives` will be called recursively on every one of the sets $\mathcal{D}'_k$, represented by the pair of lists $((r_i.P)$ $(r_1 \cdots r_{i-1}.A))$ and returned by the function `successors`; `representatives` returns the union of the result. By induction, we know that `representatives` will return a list containing some representative Q' such that $\mathcal{T}(Q) = \mathcal{T}(Q')$ when called on the set $\mathcal{D}'_k$. □

Lemma 4

$$\forall Q_1, Q_2 \in \texttt{representatives}(P, A) : \mathcal{T}(Q_1) \cap \mathcal{T}(Q_2) = \{\}$$

Proof. If P is a leaf, `representatives` can only return the one element list (P). Otherwise, `representatives` will call itself recursively on the disjoint sets $\mathcal{D}_k \cap \mathcal{T}(P, A)$. But for every $Q \in$ `representatives`(P, A), $\mathcal{T}(Q) \subseteq \mathcal{T}(P, A)$. Therefore, invocations of `representatives` on disjoint sets can never return representatives Q_1, Q_2 such that $\mathcal{T}(Q_1) \cap \mathcal{T}(Q_2) \neq \{\}$. □

Lemma 5 *If P and Q are both leaves and representatives for the same cell in $\mathcal{P}$, then P and Q contain the same pairings, and have the same size.*

Proof. Assume the lemma is false, and that there exists some pairing q that is in Q but not in P. But since the cell in transformation space that represents both P and Q is not divided any further, q must be consistent with all the pairings in P. This means that we could add q to P. This is in contradiction to the assumption that P is a leaf. □

Lemma 6 *The unpruned depth-first search tree will return every matching Q such that* `leaf`(Q) *is* `true`.

Proof. ("Unpruned" here refers to the version of the algorithm that leaves the adjoint constraints empty at all times.) This statement is essentially equivalent to the correctness of the algorithm described in Baird, 1985. □

Theorem 1 *If constraints partition transformation space into a polynomial number of cells and the function* `vbme` *runs in polynomial time, then* `all-representatives` *runs in polynomial time.*

Proof. Let L be the list returned by `representatives`. Elements of L correspond to disjoint cells of transformation space (Lemma 4); therefore `length`(L) must be smaller than the total number of cells in $\mathcal{P}$, which is polynomial in the problem size, by assumption. But every leaf in the search tree contributes an element to L. Therefore, the width of the search tree must be polynomial. It is obvious that the depth of the search tree can be at most `length(pairings)`. The computation at each node in the search tree also requires only polynomial time.

The correctness of the algorithm, i.e., the fact that it returns representatives for all the cells for which an unpruned depth first search tree would have returned a representative, has been established in Lemmas 3, 5, and 6. □

The function `all-representatives` does not exactly solve the MBM problem itself, though: the definition of MBM requires that any image or model feature is used only once within a matching, but an element in the list of pairings returned by `all-representatives` can contain several pairings that use the same image or model point.

To see how we can satisfy these additional combinatorial constraints, we introduce some additional concepts.

Lemma 7 *Every feasible matching P is the subset of some representative Q returned by* `all-representatives`.

Proof. Because of Lemma 5, it suffices to show that every feasible matching is the subset of some representative of any leaf. But this is trivial: either Q is itself a leaf, or we can add some pairings to it to arrive at a leaf (since we only have a finite number of pairings to add). □

The combinatorial constraint that no image or model feature may be used twice in a matching can be expressed as a maximal bipartite graph matching problem between image and model features (see also Cass, 1988). Let P be a set of pairings. Let the two vertex sets of the bipartite graph are given by the sets of image points in P and the set of model points in P, and the edges are given by the pairings contained in P. A largest subset of pairings that does not re-use any image or model points is a maximal bipartite matching and can be computed from P in low-order polynomial time complexity (Papadimitriou and Steiglitz, 1982).

Using these observations, we can state the following theorem.

Theorem 2 *If constraints partition transformation space into a polynomial number of cells*

$$\text{VBME} \in \mathsf{P} \Rightarrow \text{MBM} \in \mathsf{P}$$

Proof. By Lemma 7, the optimal matching is a subset of one of the matchings returned by `all-representatives`. We can run the maximal bipartite graph matching algorithm over all the representatives returned by `all-representatives` to find the optimal matching consistent with the combinatorial constraints. □

Imposing the combinatorial constraint of using each image or model feature only once is rarely necessary in practice, and often undesirable. Other combinatorial constraints and quality measures are of considerable practical interest, such as allowing multiple image features to match a single object feature, or use model boundary length accounted for in the image instead of the size of the matching. In fact, allowing multiple matches among straight line segments gives us a simple method to deal with edge fragmentation and partial occlusions of extended features.

The following, much simpler, theorem is also of some interest.

Theorem 3

$$\text{MBM} \in \mathsf{P} \Rightarrow \text{VBM} \in \mathsf{P}$$

Proof. MBM will return a maximal matching (our implementation actually can return representatives for all maximal matchings). It turns out that we only need a simpler algorithm MBM′ that returns the size of a maximal matching: *if P is compatible with* some transformation, then it is certainly necessary that MBM′($\mathcal{B}, \mathcal{M}$) returns k, where $\mathcal{B} = \{b_1 + E_1, \ldots, b_k + E_k\}$ and $\mathcal{M} = \{m_1, \ldots, m_k\}$, and that MBM′($\hat{\mathcal{B}}_i, \hat{\mathcal{M}}_i$), obtained by deleting b_i from $\mathcal{B}$ and M_i from $\mathcal{M}$, contains a matching of size $k-1$. This condition also turns out to be sufficient. □

3 Polyhedral Error Bounds

Let us now consider specific instances of the Maximal Bounded Matching (MBM) problem.

We will see that convex polygonal error bounds (linear constraints) on the position of image or model points give rise to bounds that form a convex polyhedron in transformation space for 2D rigid motions or higher dimensional affine transformations[10].

Let $\mathcal{M}$ be a finite subset of $\mathbb{R}^M$. Let the image points $\mathcal{B}$ and error bounds $\mathcal{E} = \{E_i\}$ be points and subsets of $\mathbb{R}^N$, respectively, where each subset E_i is determined by a set of

[10] Baird, 1985, gives a slightly different derivation of this fact for the 2D case and states the result in the general case without proof. Essentially equivalent is the independent result of Ullman and Basri, 1989; they use the observation that the relation between constraints induced by correspondences and the parameters of the viewing transformation is linear for testing for the consistency of several 2D images of 3D objects without explicitly computing a 3D transformation.

linear constraints $\{(d_{ik}, u_{ik})\}_{ik}$ as follows:

$$E_i = \{v : u_{ik} \cdot (v - m_i) \leq d_{ik}\} \qquad \text{where} \qquad u_{ik} \in \mathbb{R}^N, d_{ik} \in \mathbb{R}$$

In different words, each image point with error bounds is a convex polyhedron (given by the set $b_i + E_i$) whose faces are determined by the (u_{ik}, d_{ik}). Let the set of transformations $\mathcal{T}$ be the set of all transformations $T = (t, R)$, where t is a translation (vector), and R is a linear subspace of the space of $N \times M$ matrices. Let K be the total number of constraints, i.e., the sum of the number of faces of each constraint polygon.

If we pair a point b from $\mathcal{B}$ with a point m from $\mathcal{M}$ under the polyhedral error bounds E, it is easy to see that this pairing implies a collection of linear constraints on the matching transformation T. Let (u, d) be the parameters determining one of the faces of E. Requiring that T maps m into $b + E$ is then the same as requiring (for all u and d) that:

$$u \cdot (Rm + t - b) \leq d$$

But we can rewrite this as:

$$\begin{aligned} d + u \cdot b &\geq u \cdot (Rm + t) \\ &= u \cdot t + u \cdot Rm \\ &= \sum_k u^k t^k + \sum_i (u^i \sum_j R_{ij} m^j) \\ &= \sum_k u^k t^k + \sum_{ij} (u^i m^j R_{ij}) \end{aligned}$$

The last expression is formally like a dot product if we consider the vectors $\tilde{C}(u, b) = [(u^k)_{k=1,\ldots,N}, (u^i b^j)_{i=1,\ldots,N; j=1,\ldots,M}]$ and $\tilde{T} = [(t^k)_{k=1,\ldots,N}, (R_{ij})_{i=1,\ldots,N; j=1,\ldots,M}]$ elements of an $N + NM$ dimensional vector space. Observe that the R form a linear vector space–a subspace of $\mathbb{R}^{NM}$–themselves when considered elementwise. Thus, corresponding to the linear constraint associated with the pairing (b, m)

$$u \cdot (Rm + t - b) \leq d$$

is the linear constraint on $T = (t, R)$ given by

$$\tilde{C}(u, b) \cdot \tilde{T} \leq d + u \cdot m$$

This derivation goes through essentially unchanged if we impose error bounds on the image points rather than the model points; it simply reverses the roles of b and m, and the last equation reads:

$$\tilde{C}(u, m) \cdot \tilde{T} \leq d + u \cdot b$$

Sometimes we may also only have constraints on the orientation rather than the location of a features. If we represent the orientation by a unit vector o, we observe that a constraint on the orientation can also be written as a linear constraint on o as:

$$u \cdot Ro \leq d$$

This can again be converted into a linear constraint on $T = (t, R)$ since the inequality can be considered a formal dot product:

$$\sum_k 0^k t^k \sum_{ij} u^i o^j R_{ij} \leq d$$

The algorithm VBME that we need in order to apply the search algorithm from the previous section is now simply linear programming in an $N + NM$ dimensional space given K constraints (the total number of sides of all constraint polygons). We also see immediately that the space of transformations is partitioned into at most $O(K^{N+NM})$ cells through the application of all constraints.

Thus, we see that the pruned correspondence search (PCS) algorithm is a polynomial time algorithm for the the Maximal Bounded Matching problem between points in $\mathcal{B} \subseteq \mathbb{R}^N$ and $\mathcal{M} \subseteq \mathbb{R}^M$, where

- the set of transformations is given by $b = t + Rm$, $t \in \mathbb{R}^M$
- the R form a linear vector space
- constraints are given by polyhedra associated with each point either in $\mathcal{B}$ or $\mathcal{M}$

Essentially the same derivation goes through in the special case where we represent rotations and scaling in the plane by complex numbers, yielding a polynomial time algorithm for that case as well.

4 2D Recognition from 2D Images

It would be nice to be able to say more about the complexity of the recognition algorithm in specific cases. We will need to make use of the following theorem:

Theorem 4 (Megiddo, 1984) *For any fixed dimension, the linear programming problem is solvable in linear time.*

This immediately lets us infer the following theorem:

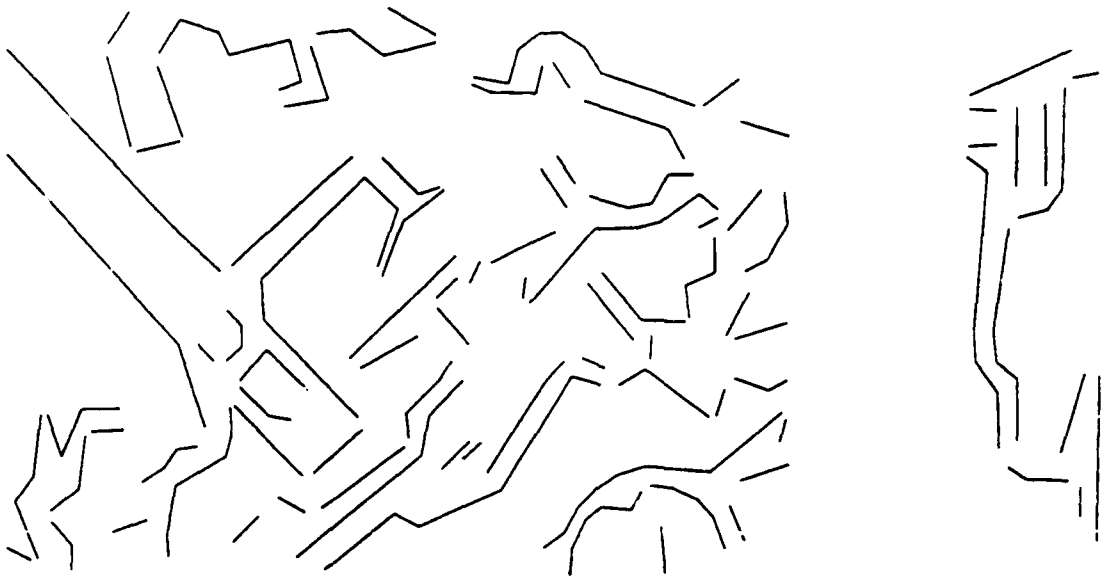

Figure 4: A matching problem. The edges of all the images were extracted from a grey level image using an implementation of the Canny edge detector and approximated by straight line segments using a splitting algorithm. Shown on the left are the edges extracted a cluttered scene with 12 widgets, on the right, the edges of the model widget. The recognition algorithm has to find the best match (in terms of model boundary length accounted for in the image) to the object on the right, allowing for translations, rotations, scaling, and occlusions. Model and image features were required to match within an error of 5 pixels.

Theorem 5 *Verification of a matching between a 2D image and a 2D model under affine transformations or rigid transformations/scale, or a 2D image and a 3D model under affine transformations and projection, can be carried out in $O(n)$ time, where n is the total number of points.*

The problem VBME in the 2D case and in the 3D case with affine transformations is linear in both the number of pairings in the matching to be verified and on the adjoint list. VBM/VMBE in the 3D case with rigid transformations is slightly harder in the worst case because of the need to intersect a quadratic surface with the polyhedron formed by the feasible transformations in transformation space (see below).

In the following theorems, for simplicity let n be the maximum of $^{\#}\mathcal{B}$ and $^{\#}.\mathcal{M}$, and assume that there is a fixed maximum number of linear constraints associated with each model point.

Theorem 6 *The worst case running time of the PCS algorithm applied to the problem of 2D recognition from 2D images under rigid transformation/scale is is* $O(n^{12.5})$.

The bound on the worst case running time follows from the fact that transformation space is divided into at most $O(n^8)$ cells by the individual constraints: there are $O(n^2)$ possible pairings, and any k linear constraints partition the four dimensional transformation space into at most $O(k^4)$ cells[11]. Each linear programming problem is at most of size $O(n)$, and the adjoint list is at most of size $O(n^2)$, giving a bound on the worst case running time of $O(n^{11})$. The bipartite graph matching costs at most $O(n^{2.5})$, but we already have counted in $O(n)$ time for the verification of each search node via linear programming. □

This bound is very loose. First, the algorithm may actually only explore a fraction of the cells in transformation space. Furthermore, both the linear programming algorithm and the maximal bipartite matching algorithm can be interleaved with the search so that partial results are taken advantage of at deeper levels: such techniques are already known to reduce the overall complexity of the algorithm further in some cases (Baird, 1985).

Notice that any transformation space sweep algorithm that enumerates all cells for the same problem has worst case *and* average case complexity of at least $\Omega(n^9)$, since the number of cells in the arrangement formed by the linear constraints on feature locations has size $\Omega(n^8)$ in the worst and average case, and the cost of picking a transformation and computing a corresponding matching is $\Omega(n)$.

In order to be able to say anything about the average case performance of the PCS algorithm, observe that the adjoint-based pruning step in the PCS algorithm incurs at most an additional overhead of $O(n^2)$ over any analogous correspondence based algorithm. This observation lets us carry over average case analyses such as the ones found in Baird, 1985, Grimson, 1988, and Grimson, 1989, to the corresponding pruned algorithm.

5 3D Recognition from 2D Images

For the recognition of rigid 3D objects from 2D images, the restrictions on the matrices R cannot be expressed in linear form anymore. The VBME algorithm now requires the simultaneous solution of linear inequalities together with a set of quadratic equalities whose number is independent of problem size. One approach is to embed the manifold formed by all rotations and scaling matrices in a linear space, solve the verification problem in the larger space using linear programming, and then determine whether the intersection between the manifold and the polyhedron containing the feasible transformations is non-

[11] This fact can be demonstrated either algebraically or inferred using the Vapnik Chervonenkis (VC) dimension.

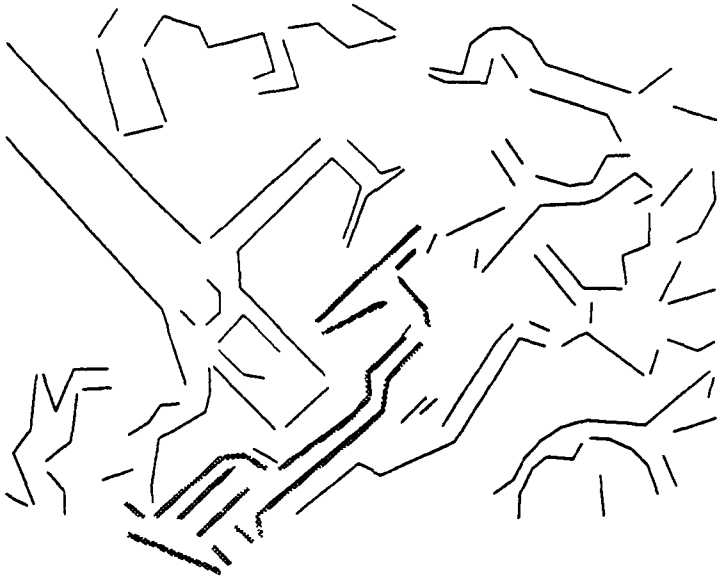

Figure 5: The solution to the matching problem. The figure shows an optimal transformation of the model onto the image obtained using the PCS algorithm. Previous algorithms, such as heuristic search termination or incompletely pruned algorithms, either did not find the optimal solution, or could not be run to completion because of the combinatorial explosion of the number of possible matchings.

empty. The cost of this intersection test is bounded by $O(k^d)$, where d is the dimension of the space containing the rotations, and k is the number of linear constraints forming the surface of the polyhedron. Although k can in principle be as large as n^2, it can be large only very rarely, and we will therefore assume that the cost of the intersection test is constant in an amortized analysis.

Putting all these results together, we can bound the worst case complexity of the PCS algorithm when applied to 3D recognition. Transformation space in this case is five dimensional ($\mathbb{R}^2 \times R^3$, where $\mathbb{R}^2$ is the two dimensional Euclidean space of translations, and R^3 is the three dimensional manifold of rotations), meaning that the $O(n^2)$ constraints can give rise to at most $O(n^{10})$ cells. Taking into account the $O(n)$ cost for linear programming, the $O(n^2)$ cost for maintaining and testing the adjoint list, and the $O(n^{2.5})$ cost for the maximal bipartite graph matching this yields a bound of $O(n^{14.5})$ on the worst case time complexity of the algorithm for 3D recognition from 2D data under rigid transformations and convex polygonal error bounds. For the same reasons as in the 2D case, this is only a loose bound.

In the average case, again, assuming constant time for the intersection test, the algorithm will perform at most $O(n^2)$ worse than any existing correspondence algorithm. To see what average case complexity this works out to in practice, consider the case of recognition from 2D features with location and orientation. In the complete absence of error, we have to consider $O(n^4)$ pairs of image and model features and solve a set of linear inequalities in time $O(n)$ in order to find a maximal matching (this is simply the complexity of alignment algorithms). The same considerations carry over to an average case analysis for sufficiently small error bounds. The additional overhead of the PCS algorithm is $O(n^2)$, giving a loose bound on the total average case complexity of $O(n^7)$ for 3D object recognition from 2D images under the above assumptions.

6 Discussion

We have seen that under weak assumptions the Maximal Bounded Matching (MBM) problem is at most polynomially harder than the corresponding verification problem VBME, and, conversely, that a polynomial time algorithm for MBM implies the existence of a polynomial time algorithm for the verification problem VBM.

There are three important properties of the recognition problem as defined above that make it possible to find tractable algorithms to solve it:

1. The bounded error model makes the problem discrete.
2. The "quality measure" (size of a matching) is invariant under permutations of image or model features within a matching.
3. Transformation space has a fixed, bounded dimensionality, and error bounds on image or model features induce subsets in tranformation space that have a simple structure.

All three of these properties are needed. For example, for a quality measure whose dependence on the matching itself is complicated, there is obviously no reason to expect polynomial time performance of a recognition algorithm: as we have observed already above, even given a single transformation that aligns n image and model points, the number of possible matchings that are geometrically consistent with the transformation can be as large as $n!$, and a general quality measure could assign arbitrary values to each of the possible matchings. Or, if we do not fix the dimensionality of transformation space (as in recognition-with-parts problems, where the number of possible parts is unlimited), a small number of location constraints can give rise to a large number of distinct and geometrically feasible matchings.

These properties of the recognition problem have also been taken advantage of in the transformation space sweep (TSS) algorithms for recognition described by Cass, 1990. An important difference between such algorithms and the correspondence based algorithm presented here is that the partition of transformation space used in the TSS algorithm, the arrangement induced by considering all location constraints simultaneously, is finer than actually needed for finding a maximal matching.

For example, each individual location constraint may give rise to a number of disconnected sets in transformation space (as it does in the case of 3D recognition under rigid transformations). A straightforward application of the TSS algorithm has to enumerate and test separately each of the disconnected components of such sets. In contrast, the PCS algorithm only relies on the existence of a verification algorithm to determine whether the intersection of sets of transformation space given as sets of pairings are non-empty: the topological properties of the sets of feasible transformations involved do not matter (and sets with complicated topology can still be comparatively easy to test for intersection with one another).

Another important distinction between the PCS algorithm and transformation space sweep algorithms is that PCS makes direct use of combinatorial constraints. This means that the PCS algorithm only computes a subset of those representatives that are actually needed for finding a maximal matching.

When compared with previous complexity results for correspondence search techniques in vision, the existence of the PCS algorithm demonstrates that the absence of unique feature labels and the presence of clutter and occlusions in visual recognition does not necessarily lead to an exponential behavior for correspondence search algorithms as has been conjectured by Grimson, 1989.

Another important difference between correspondence search algorithms as described by Grimson and Lozano-Perez, 1983, and the algorithm described in this paper is that their algorithm has expected polynomial time behavior only if a match is known to exist in between the image and the model; the time required to determine absence of a sufficiently large match between image and model for the unpruned algorithm requires exponential time even in the expected case. The PCS algorithm described in this paper has worst case (and, hence, expected) polynomial time complexity in both cases.

The pruning method employed by PCS is a relatively low-cost scheme for pruning a depth-first tree search for a maximal matching; for convex polygonal location constraints, in the worst case, it will run $O(mn)$ (where m is the number of model points and n is the number of image points) slower than any equivalent correspondence based algorithm without the pruning step. In practice, we find that the pruning step actually improves the overall performance of the algorithm in the case of moderate or large error bounds, since it avoids re-exploration of transformations that have already been tested; existing,

incompletely pruned algorithms will explore too many search nodes when error bounds are large because of the combinatorial explosion of the number of geometrically feasible matchings.

Existing heuristics can be incorporated into the PCS algorithm to improve its performance while still guaranteeing worst-case polynomial complexity. In particular, the technique of Grimson and Lozano-Perez, 1983, uses a test for pairwise consistency to exclude impossible matchings quickly, and the same technique can be incorporated into the pruned algorithm described in this paper. Furthermore, all the branches at a node in the search tree can be explored in parallel, making the algorithm attractive for a parallel implementation.

For the verification step, linear programming methods are particularly desirable, since they are provably efficient, and good implementations exist. If the non-linearities in the verification step are of a simple enough form and fixed in number (as, for example, in the recognition of rigid 3D objects from 2D data), efficient verification algorithms may still be formulated, as described above.

Acknowledgements

The author would like to thank Tao D. Alter, Todd Cass, Eric Grimson, Leonidas Guibas, Robert S. Thau, Tomaso Poggio, and Shimon Ullman for numerous discussions about recognition, and their suggestions and comments on the paper.

References

Alt H., Mehlhorn K., Wagener H., Welzl E., 1988, Congruence, Similarity, and Symmetries of Geometric Objects., *Discrete and Computational Geometry.*

Baird H. S., 1985, *Model-Based Image Matching Using Location*, MIT Press, Cambridge, MA.

Breuel T. M., 1989, Adaptive Model Base Indexing, In *Proceedings: Image Understanding Workshop*, pages 805–814, Los Angeles, CA, Morgan Kaufmann, San Mateo, CA.

Breuel T. M., 1990, Indexing for Visual Recognition from a Large Model Base, A.I. Memo No. 1108, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Cass T. A., 1988, Robust Parallel Computation of 2D Model-Based Recognition, In *Proceedings Image Understanding Workshop*, Boston, MA, Morgan and Kaufman, San Mateo, CA.

Cass T. A., 1990. Feature Matching for Object Localization in the Presence of Uncertainty. A.I. Memo No. 1133, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Edelsbrunner H., 1987. *Algorithms in Combinatorial Geometry*. Springer Verlag.

Grimson W. E. L., Huttenlocher D. P., 1989. On the Verification of Hypothesized Matches in Model-Based Recognition. A.I. Memo 1110, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Grimson W. E. L., Lozano-Perez T., 1983. Model-Based Recognition and Localization From Sparse Range or Tactile Data, Technical Report A.I. Memo 738, MIT.

Grimson W. E. L., 1988. The Combinatorics of Object Recognition in Cluttered Environments using Constrained Search, Technical Report A.I. Memo 1019, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Grimson W. E. L., 1989. The Combinatorics of Heuristic Search Termination for Object Recognition in Cluttered Environments., A.I. Memo No. 1111, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Khachian L. G., 1979. A Polynomial Algorithm for Linear Programming *(translation)*. *Soviet Math Doklady*, 20.

Megiddo N., 1984. Linear Programming in Linear Time when the Dimension is Fixed. *J. Assoc. Comput. Mach. (USA)*, 31(1).

Norton J. P., 1986. *An Introduction to Identification*. Wiley.

Papadimitriou C. H., Steiglitz K., 1982. *Combinatorial Optimization*. Prentice Hall.

Rees J. A., Clinger W., 1986. Revised^3 Report on the Algorithmic Language Scheme. *ACM Sigplan Notices*, 21(12).

Ullman S., Basri R., 1989. Recognition by Linear Combinations of Models. A.I. Memo No. 1152, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.